LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-682663

# ROPE: Recoverable Order-Preserving Embedding of Natural Language

D. Widemann, E. X. Wang, J. Thiagarajan

February 11, 2016

## Disclaimer



This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

# ROPE: Recoverable Order-Preserving Embedding of Natural Language

**Eric X. Wang**
Lawrence Livermore National Laboratory
wang73@llnl.gov

**David P. Widemann**
Lawrence Livermore National Laboratory
widemann@llnl.gov

**Jayaraman J. Thiagarajan**
Lawrence Livermore National Laboratory
jayaramanthi1@llnl.gov

## Abstract

We present a novel Recoverable Order-Preserving Embedding (ROPE) of natural language. ROPE maps natural language passages from sparse concatenated one-hot representations to distributed vector representations of predetermined fixed length. We use Euclidean distance to return search results that are both grammatically and semantically similar. ROPE is based on a series of random projections of distributed word embeddings. We show that our technique typically forms a dictionary with sufficient incoherence such that sparse recovery of the original text is possible. We then show how our embedding allows for efficient and meaningful natural search and retrieval on Microsoft's COCO dataset and the IMDB Movie Review dataset.

## 1 Introduction

Distributed text representations have received significant attention lately [1, 2]. Representations such as *word2vec* [1] and *GLoVE* [2] encode words in high dimensional vector spaces where relative spatial proximity and geometry encode significant syntactic and semantic information. These embeddings in turn enable powerful order-sensitive natural language processing (NLP) algorithms, and have been applied with success in fields such as sentiment analysis [3], sentence completion [4] and language and caption synthesis [5, 6].

Much recent NLP research using distributed word representations ingests a series of word vectors directly [7, 8, 9]. This typically requires setting the number of words seen by the model *a priori*, a choice that can have important effects on the behavior of the model. We propose an alternative approach to language understanding and sentence-level feature extraction that relies on tools and concepts developed for compressive sensing (CS) and sparse recovery. Our approach, which we call ROPE for Recoverable Order-Preserving Embedding, has several advantages

1. Sentences of arbitrary lengths (up to a preset-maximum) are encoded into a fixed feature size.
2. The Euclidean metric in the distributed representation space preserves semantic meaning. Sentences that have similar meaning and structure are close to each other in the $L_2$-norm. This makes it possible to search for similar semantic meaning across variable length sentences.
3. Given a set of learned word vectors Encoding sentences via ROPE is highly efficient, requires no parameters to learn, and can be parallelized readily.
4. Text can be recovered/synthesized from the embeddings via well known sparse recovery techniques by taking a dense vector in $\mathbb{R}^N$ to a high dimensional sparse vector denoting non-zero dictionary entries.
5. Embedding text via ROPE has extremely low computation overhead, allowing for on-the-fly creation of passage-length feature vectors.

The primary application of ROPE is as a semantically meaningful feature extraction tool for text. This potentially allows variable-length text passages embedded with their word-ordering by ROPE to be ingested by a wide variety of

model architectures that were previously unavailable to NLP researchers such as convolutional neural networks, linear and kernel based classifiers, and clustering algorithms.

An outline for this paper is as follows. First we describe the compressive sensing (CS) and demonstrate how CS can be employed for sparse recovery of text. We next show that distributed embeddings, specifically the vectors learned by the popular word2vec, also yield meaningful and recoverable basis. These results are extended to give Recoverable Order Preserving Embeddings (ROPE). Results for ROPE are shown that demonstrate its efficacy for document retrieval using the Microsoft COCO dataset [10]. We conclude by discussing challenges and future work.

## 2 A Brief Review of Compressive Sensing

Compressive sensing concerns the recovery of a sparse vector given a real valued observation vector and an appropriately designed sensing matrix. Consider a real valued signal $X \in \mathbb{R}^D$. Any signal in $\mathbb{D}^V$ can be represented by a basis $\Psi$ consisting of $\mathbb{R}^D$ vectors. The signal $X$ is $K$-sparse is it can be represented by a subset of $K$ columns of $\Psi$. Let $S \in \mathbb{R}^V$ denote the weighting coefficients such that

$$X = \Psi^T S, \tag{1}$$

where $S$ has $K$ non-zero elements. Further, let $\Phi \in \mathbb{R}^{P \times D}$ denote a fixed measurement matrix. Then the compressive measurement $Y \in \mathbb{R}^R$ is

$$Y = \Phi X = \Phi \Psi^T S = \Theta S \tag{2}$$

where $\Theta = \Phi\Psi^T$ is an $R \times V$ matrix. The basis $\Psi$ need not be orthogonal, and in fact can be highly overcomplete [11], a fact we will exploit in this work. We construct the sensing matrix $\Phi$ by sampling its elements from $N(0, 1/R)$; with overwhelming probability this construction insures that the matrix $\Theta$ satisfies the restricted isometry property (RIP),

$$(1 - \delta_K)||S||_2 \leq ||\Theta S||_2 \leq (1 - \delta_K)||S||_2 \tag{3}$$

for $\delta_K \in (0, 1)$ [12, 11]. Given $Y$ and an RIP-satisfying $\Theta$, the sparse vector $S$ can be recovered exactly [12], typically with either an L1 regularized recovery or a greedy matching pursuit type algorithm. More recent work has shown Bayesian methods also work well.

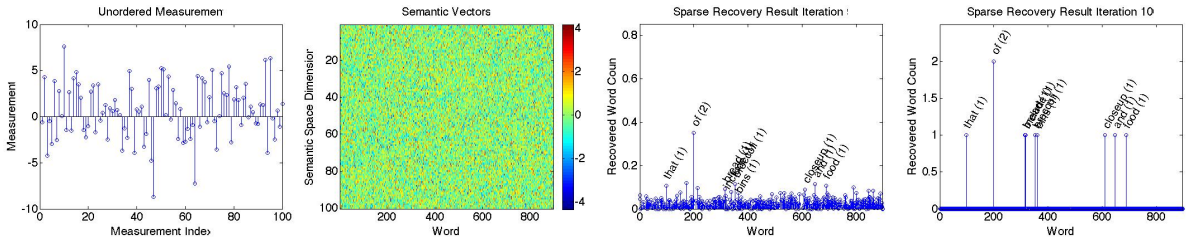
## 3 Sparse Recovery of Unordered Text

We begin by discussing a simple example of unordered text recovery. Consider a large vocabulary of $V$ words, and let $S$ be a $K$-sparse $V$-dimensional vector of word frequencies. If $S$ encodes a short passage (for example a sentence) then $S$ will naturally be sparse. By observing the compressed measurement $Y = \Phi S$ and the RIP-satisfying measurement matrix $\Phi$, we can perform CS inversion [13] to recover $S$ with low error without knowing the value or position of the non-zero elements *a priori*. Any CS inversion algorithm can be used to recover the original signal. We choose to use the Variational Bayes Relevance Vector Machine (VB-RVM) as detailed in [14] for all inversions demonstrated in this paper.

In Figure 1 we show the process of recovering the phrase "closeup of bins of food that include broccoli and bread". The elements of the RIP-satisfying matrix (shown in Figure 1(b)) are sampled i.i.d. from a zero-mean Gaussian distribution. As these are demonstrations, for computational speed we culled $90\%$ of the dictionary, resulting in approximately $V = 850$ unique tokens (and their corresponding vectors) being considered in the projection and recovery. Following [15], we set $N = 100 > Klog(V)$ with $K = 10$ resulting in a 100-dimensional measurement (Figure 1(a)).

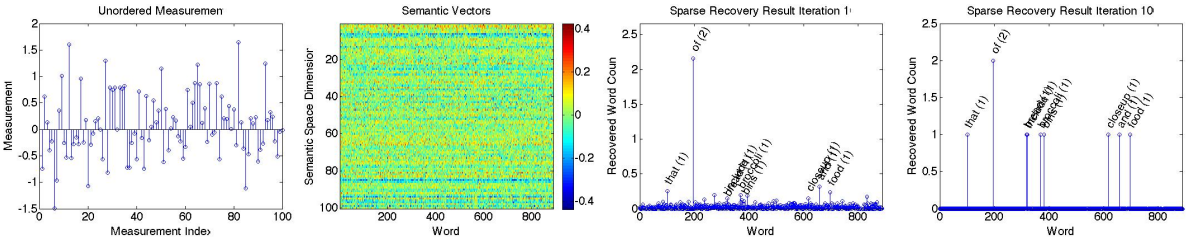
That the original sparse signal (encoding text or otherwise) can be recovered with high fidelity via a RIP-satisfying random projection is well known. However, suppose we train a set of $D$ dimensional distributed word representation on a corpus containing $V$ unique words. Let $B$ be a $D \times V$ dimensional matrix whose columns are the individual word vectors. A surprising result is that if we simply use the $B$ as the sensing matrix instead of the RIP-satisfying matrix $A$, we can still recover $S$ with no apparent degradation in performance. We show this in Figure 2 with $D = N = 100$. Here the only difference to the preceding case is that a semantic text embedding is trained beforehand to learn the structure shown in Figure 2(b). Note that significantly more structure is present in the semantic projections of 2(b) than in the random matrix of 1(b).

An advantage of of using the distributed word vectors over the RIP satisfying embedding is that semantic information is included in the sentence embedding via the distributed representation, thus the sentences "a dog ran across a park" and "a puppy ran across a park" will be proximate to one another due to the semantic closeness of "puppy" and "dog". We believe that the ability to recover the sparse signal from the highly structured semantic embeddings is due to the fact that, while $B$ does not necessarily satisfy RIP, its columns exhibit sufficient incoherence such that they satisfy the relaxed recovery conditions of [16].




(a) (b) (c) (d)

Figure 1: Sparse recovery of unordered text via a RIP satisfying embedding. (a) Observed measurement. (b) RIP satisfying matrix. (c) and (d) show VB-RVM recovery at 10 and 100 iterations, respectively.




(a) (b) (c) (d)

Figure 2: Sparse recovery of unordered text via word2vec vectors. (a) Observed measurement. (b) The word2vec vectors (columns). (c) and (d) show VB-RVM recovery at 10 and 100 iterations, respectively.

## 4 Recoverable Order-Preserving Embedding (ROPE)

Based on the previous section, ordered recovery of text requires only minor changes to our embedding algorithm. In this section, we will redefine the sparse representation $S$ and the measurement matrix $A$.

Instead of a word frequency vector encoding all the words in a string, suppose we encode each word by a one-hot $V$-dimensional vector. We will consider up to $K$ consecutive words. These one-hot vectors are concatenated in the order the words appear into an $M$-dimensional $K$-sparse vector, where $M = KV$. $K$ is chosen to be the maximum length text passage we expect to encounter. Typical sentences will be shorter, and thus have sparsity less than $K$.

To construct the projection matrix $\Theta \in \mathbb{R}^{N \times KV}$, we first construct the sparse basis $\Psi$ by loading word embeddings $B$ onto its main diagonal with zeros elsewhere. Thus, $\Psi$ is a $kD \times kV$ block diagonal matrix. The RIP-satisfying sensing matrix $\Phi$ has dimensionality $R \times kD$ and has elements sampled from $N(0, 1/R)$

This construction allows for the usage of a word at different word positions to be identified. Each copy of the word embeddings are completely incoherent with the other copies due to the block diagonal structure. Within each copy of the word embeddings, there is significant redundancy, as the typical number of words $V$ is significantly larger than the word embedding dimensionality $D$. Happily, the properties of RIP-satisfying matrices extends to this exact situation [11], and the general geometry of the semantic space, encoding the rich relationships among concepts is preserved.

Below in Figure 3 we show an example of ordered recovery. We set $N = 500$, $D = 100$, and consider an embedding of up to 15 words, resulting in a sensing matrix consisting of 15 randomly projected copies of the *word2vec* vectors with $M = 15V$.

The projection matrix $\Theta$ is shown in Figure 3(b). We show the recovered sentence in Figure 3(c) where again the vertical lines denote the repeated dictionaries. Note that within each dictionary, only one word has a non-zero value.

## 5 Results

### 5.1 Caption Retrieval via ROPE

Recently, Microsoft released the Common Objects in COntext dataset (COCO) [10] dataset, comprising over $300,000$ images. Each image is partially segmented, with labels for the segments. 80 unique labels are considered. Additionally,

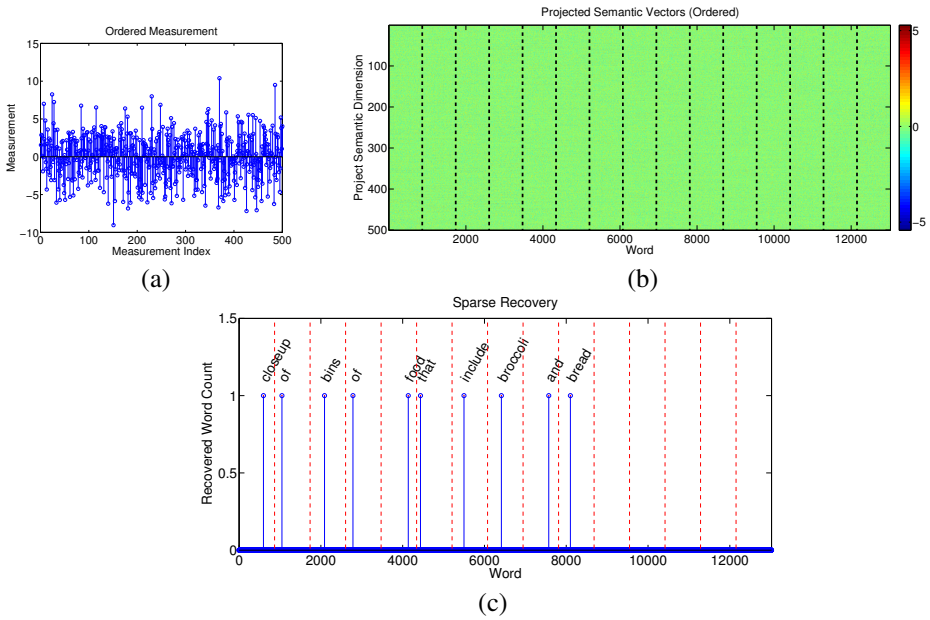



Figure 3: Sparse recovery of ordered text via word2vec vectors. (a) Observed measurement. (b) Measurement matrix constructed by randomly projecting copies (denoted by vertical lines) of the word2vec vectors via RIP satisfying matrices. (c) The recovered sentence.

each image is associated with 5 full English captions. For our experiments, we used the $80,000$ image training set from the COCO 2014 release and the $40,000$ image validation set as our out-of-set testing dataset.

We demonstrate two different caption retrieval methods via ROPE on the MS COCO dataset. In the first demonstration, we show results using previously unseen out-of-set examples from the validation set as queries. In the second demonstration, we show qualitative results using previously unseen user generated (arbitrary) text as queries. In both cases, results are ranked by $L_2$ distance in the embedded space. Note that we show the images associated with the retrieved captions for illustrative purposes only, and image feature vectors are not considered in our search results. For the COCO examples, we used embeddings capable of considering up to $50$ word long passages in order to cover the longest caption in the COCO training set.

### 5.2 Querying with user-generated out-of-set captions

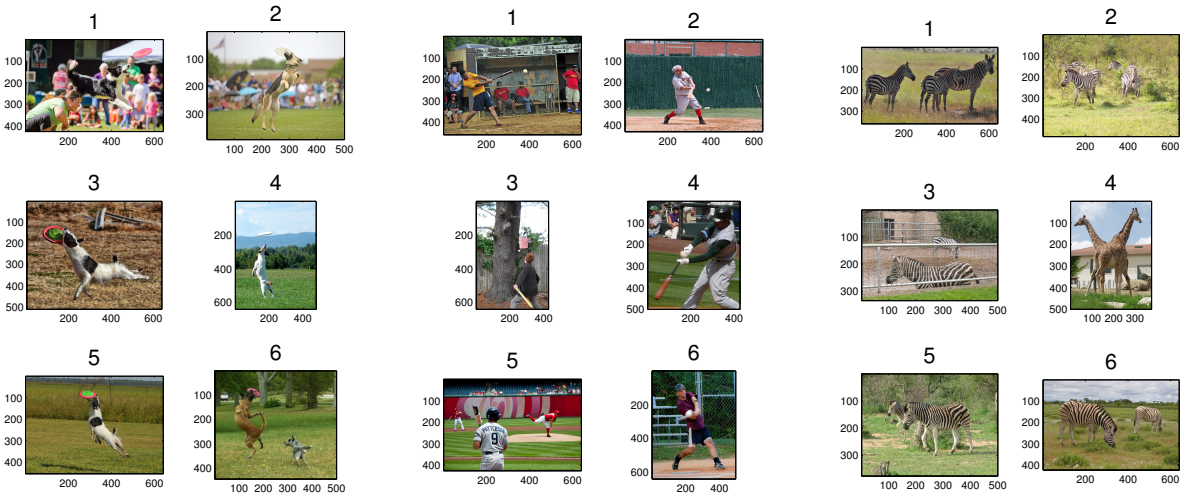
Given a query of length $K$, the distributed representations $B$, and the random projection matrices $R_k$, ROPE maps the query into its embedding space as

$$V_{query} = \sum_{k=1}^{K} R_K B_{c_k} \tag{4}$$

where $c_k$ denotes the dictionary position of the $k$th word in the query and $B_c$ is the $c$th column of $B$.

Below in Figure 4 we show several examples of searches using user-created queries. For each query, we show the 5 closest captions and their associated images. These results demonstrate that while ROPE is not a language model, the ordered-preserving embedding space it generates is quite rich in an NLP sense. For example, in Figure 4(a) the query includes a passage "runs across a park" that does not appear as a previously observed phrase in the training set; however, the embeddings are still able to strongly capture the concept of a dog catching a frisbee. In Figure 4(b), the query "a man hits a baseball" does not return any matches with the word "hit"; instead the search returns results with "swinging a bat". In Figure 4(c) the embedding is able to discern a the word "some"denotes a plurality, and responds with pictures of mutliple zebras. The error made (by the inclusion of giraffes) in 4(c) can be explained by the semantic similarity of giraffes and zebras in the semantic space.

Notably, the results and queries all differ in length, and many differ in linguistic structure. This shows that, given sufficient data to create the embedding space, ROPE captures significant semantic and syntactic information. While



Query: a dog runs across a park to catch a frisbee
Answer 1: A dog leaps in the air to catch a frisbee
Answer 2: A dog leaps in the air to catch a Frisbee.
Answer 3: A dog jumps in the air to catch a frisbee.
Answer 4: a dog jumping in the air to catch a frisbee
Answer 5: A dog jumping in the air to catch a Frisbee.
Answer 6: a dog jumps in the air to catch a frisbee

(a)

Query: a man hits a baseball
Answer 1: A man swinging a bat towards a baseball.
Answer 2: A man swinging a bat towards a baseball.
Answer 3: A boy swings a bat at a pinata.
Answer 4: A man swinging a bat at a baseball.
Answer 5: A man pitches a ball to the a batter
Answer 6: A man swinging a bat towards a ball.

(b)

Query: some zebras standing in a field
Answer 1: Three zebras standing in a field of grass.
Answer 2: Several zebras grazing in a lush green meadowland.
Answer 3: Two zebras standing in a fenced zoo enclosure.
Answer 4: Two giraffes standing in a grassy enclosure.
Answer 5: Two zebras running in a field past brush
Answer 6: two zebras standing in a grassy field grazing

(c)

Figure 4: Sample user generated query results.

ROPE does not allow a machine to understand a sentence per se, it does create a vector space where abstract concepts (encoded in structured passages of words) can be compared to one another in an intuitive and straightforward manner.

### 5.3 Querying with validation out-of-set captions

We use the captions and labels from Microsoft COCO 2014 validation dataset as a test set. The test set contains $200,819$ captions, each with several labels corresponding to segments in its associated image. For this experiment, we treated the segment labels as ground truth.

For each test caption, we retrieved the $50$ closest matches in the embedded space, and computed the Top-$N$, $N = 50$, precision and recall scores for two different matching criteria: Single Match - a "success' if the returned result shares at least 1 label with the query labels; and Half Match - a "success" if the returned result matches at least half of the query image's labels. In the case of an odd-numbered query labels, we use the next largest integer. For queries with single labels, the match is all-or-nothing. Below in Figure 5 we show the mean Top-$N$ precision scores for ROPE as well as the random baseline.

As expected, the Single Match criteria results in improved precision but significantly reduced recall compared to the more strict Half Match criteria. The random baseline recorded precision scores near $0.1\%$.

## 6 Results on IMDB Movie Review Dataset

The IMDB Movie Review dataset is a collection of 25,000 movie reviews, of which 12,500 are positive and 12,500 are negative [17]. Each review is partitioned by its sentences and then each sentence is embedded via ROPE. There are approximately 600k sentences in total and the sentence variance in structure and length is much greater than that of the Microsoft COCO 2014 dataset. A sentence is chosen at random and a query is performed to find its closest neighbors. The performance of ROPE for semantic search and retrieval on the IMDB dataset is mixed. There are examples of ROPE successfully returning semantically meaningful results.

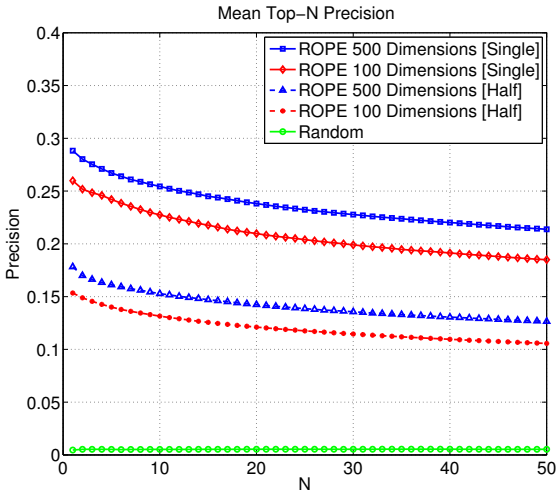



Figure 5: Experiemental Results using Microsoft COCO 2014 Validation set as queries for ROPE embeddings of 100 and 500 dimensions.

**Query: "This is one of the most beautiful and refreshing films that I have seen in some time."**

The top-4 returned sentences are:

1. This is one of the most important and powerful films I have seen in quite some time.
2. This is one of the most brilliant movies that I have seen in recent times.
3. This is one of the most life-affirming films that I have ever seen.
4. It is one of the most beautiful documentaries that I have ever seen and one of the most spiritual.

**Query: "She and Dalton have wonderful chemistry and their scenes together are pure delight."**

The top-4 returned sentences are:

1. She and Matthau have great, unforced chemistry in their scenes together.
2. Cagney and Blondell have excellent chemistry and their scenes go off really well.
3. I love seeing Harlow and Gable together and in this film they are simply wonderful.
4. Channing and Amanda have amazing chemistry and were absolutely wonderful together.

The are synonyms in the returned results that would be missed if with a standard key-word search. There are examples of ROPE failing to return meaningful results too.

**Query: "And his mother...how the hell does she figure something like that out?"**

The top-3 returned sentences are:

1. But Fontaine is one of those gals who has eyes only for money, and the man standing between her and it is transparent, so that she doesn't even notice or care what he looks like, she looks through him and sees what she really wants and goes for it.
2. Even that death was stupid because the statues tooth went through his mouth and hangs there like that will support it and there is a scene when a goth girl loses her contacts doesn't find them, and seems like she doesn't need the.
3. And if you still can't figure out which one he is, here's a hint: The auteur and his character have the same middle name.

The sentences in the above example are long and complex. While it is clear how their sentence embeddings are formed, it is not clear why these sentences would be semantically close to our query. We have numerous examples of ROPE exhibiting this sort of behavior on long, complex sentences.

## 7 Computational Aspects of ROPE

The computational steps for performing searching and retrieval with ROPE are as follows:

1. A word embedding is created for the corpus using *word2vec* or *GLoVe*.
2. The corpus is tokenized by at the sentence level.
3. Each sentence is embedded using the above word embedding.
4. The search query is embedded using the same projections matrices as were used for the sentences above.
5. The Euclidean distance is computed between between the embedded query and all of the embedded sentences.
6. The $k$-closest sentences are returned.

The most time consuming component above is step 3, embedding the sentences. Computationally, this is a trivial task in which a sequence of vectors are added; however, the large number of sentences in the corpus translate into long run-times. For example, embedding the 600k IMDB movie review sentences on a single node of our Surface cluster required approximately 5 hours. Once the embedding is done, then it can be saved and queried against repeatedly. In this sense, the sentence embedding step of ROPE is analogous to the indexing step for key-word search algorithms. In order to do key-word search on a corpus, the corpus must first go through the time consuming process of being indexed. Second, the sentence embedding step of ROPE is parallelizable. The embedding for each sentence can be independently. Preliminary tests with parallel codes have shown that the embedding step scales linearly with the number of processors.

## 8 ROPE vs Key-word Search

The main competitor for search and retrieval with ROPE is key-word search. In key-word search, a look-up is performed for each word in the query using the index for the corpus. This look-up returns a set of documents that use the query word. A set is created for each word in the query. If an *and* search is performed, e.g. dog + bone, then the intersection of these returned document sets is computed and each document therein is scored. For example, the documents that contain the word "dog" are in one set and the documents that contain the word "bone" are in another. These two sets are then intersected. If an *or* search is performed then the union of these two is taken.

The benefits of key-word search are:

- Numerous tools for indexing and querying datasets
- Returned results are easily interpreted and very well understood

The cons of key-word search are:

- It does not use word embeddings so there is no notion of words being semantically close. A search for dog will not return results that contain the word puppy in-lieu of dog. This rules out synonym searches too. This in contrast to ROPE.
- There is no structure for the query. With key-word search, the order and the relationship between the query words is irrelevant. In contrast, ROPE specifically embeds sentences in a manner that captures word position information.

Overall, ROPE and key-word search can be seen as complimentary algorithms. There are cases in which key-word search will fail but ROPE will succeed because it uses a word embedding that captures semantic meaning. Similarly, there are examples of long sentences in which ROPE returns poor results that are hard to interpret yet key-word search gives good results that are understood. The complimentary nature of ROPE and key-word search gives reason for using both techniques in tandem.

## 9 Metrics

There is a large question and answer dataset that is used to test how well a word embedding performs on analogies. For example, a word embedding's performance on semantic analogies can be tested by asking questions like *"man is to king as woman is to what?."* Similarly, syntactic questions can be a tested by asking questions such as, *big is to bigger as tall is to what?*." In our experience, the *GloVe* algorithm tends to perform better than *word2vec* on semantic analogies whereas *word2vec* tends to perform better on syntactic analogies. Both of these embedding techniques have problems with homonyms. When a word has two different meanings it will still get mapped to a single word vector. We are investigating a technique for modifying *GloVe* to handle homonyms.

For sentence level embeddings, there is no standard metric for measuring their performance. The Bleu score is used for comparing sentences from automated machine translation. For example, a machine will translate a sentence from Russian to English. The translated sentence will then be compared against a set of human translated sentences by essentially counting the number of words that the automated translation has in common with the human generated set. The Bleu score is slightly more complex than this, but that is the basic idea. We could do something similar for ROPE. However, the formula for the Bleu score does not take into account synonyms. Synonyms are one the major advantages of ROPE. In ROPE space, the sentences *"The puppy ran."* and *"The dog trotted."* are fairly close, yet the Bleu score for these two sentences would be zero if we were to remove stop words. We are searching for datasets that have sentences grouped or labeled by semantic meaning. The language translation datasets with multiple outputs sentences may be a good candidate for testing ROPE's performance in this regard.

## 10 ROPE Challenges

There are problems with ROPE that involve extraneous words, sentence length and negation. For extraneous words, ROPE uses projection matrices to map each word in the sentence. This allows for encoding the position of each word along with its word vector and enables order-preserving sentence recovery. However, the downside of this is that often these"filler" words push other sentence words into different projected spaces than they would otherwise be. This causes the sentence embedding to be very different than what it would be without filler words. One solution to this, is to remove the projection matrices from the ROPE algorithm. The cost of doing this is that ROPE will no longer be order-preserving. However, ROPE will be more robust against extraneous words.

We have also found that ROPE performs poorly on long sentences and does not handle negation well. Often, long sentences will discuss multiple concepts. The ROPE embedding is mapping these multiple concepts into one vector. It may not be possible to capture multiple concepts with single vector. There is research in which sentences are first converted to tree structures such as constituent trees and then processing is done on the sentence and its constituent tree [18]. The constituency tree structure allows for complex sentences to be broken down into more manageable pieces and negation can be modeled in the tree structure. Figure 6 below, gives a the constituency tree for this long sentence containing negation, *"They were not actively developing nuclear weapons, but they contemplated it."*

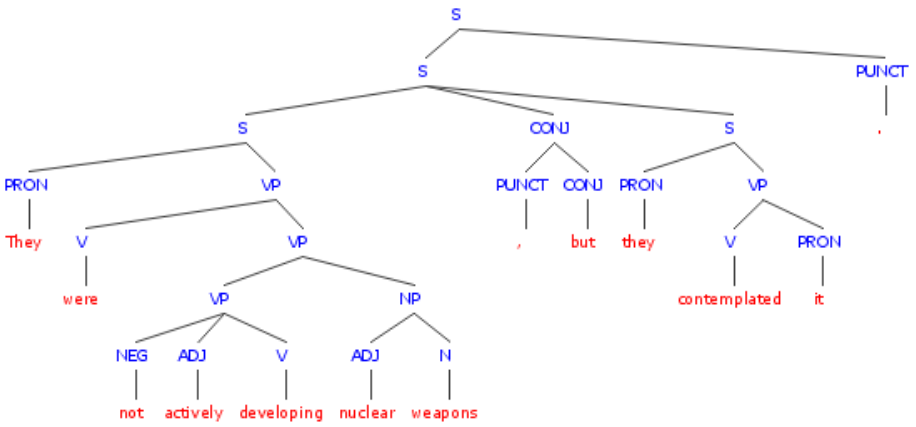



Figure 6: Constituency tree for a moderately complex sentence with negation.

## 11 Future Work

Above, we discussed applications of word embeddings for search and retrieval. However, word embeddings coupled with recurrent neural networks and memory networks [19] are providing state-of-the-art results in a wide range of NLP tasks. Currently, our research program is incorporating these algorithms for sentiment analysis, machine translation, search and retrieval. I will briefly outline our approach for each task.

For sentiment analysis, we are ingesting large datasets of labeled data for training purposes. This includes the IMDB Movie Review and the Stanford Treebank datasets. We are constructing recurrent neural networks that use LSTM and GRU modules for recurrency and an embedding layer to learn word vectors. The embedding layer for neural networks has been an interesting development for word embeddings. Instead of mapping words to some *a priori* word embedding, the embedding can be learned for the task at hand. This works by back-propagating the error to the word vectors themselves and making updates there.

An unrolled, two-recurrent, LSTM network as described above can be visualized in Figure 7. A passage of text is tokenized at the word level and then the sequence of words are input into the network. The network has memory states (hidden) that change based on the input sequences words. This allows for the network to learn complex rules and behaviors. In the case of sentiment analysis, the final output is a single number, represented by 1 box. Networks such as this coupled with parse trees are giving state-of-the-art results for sentiment analysis.

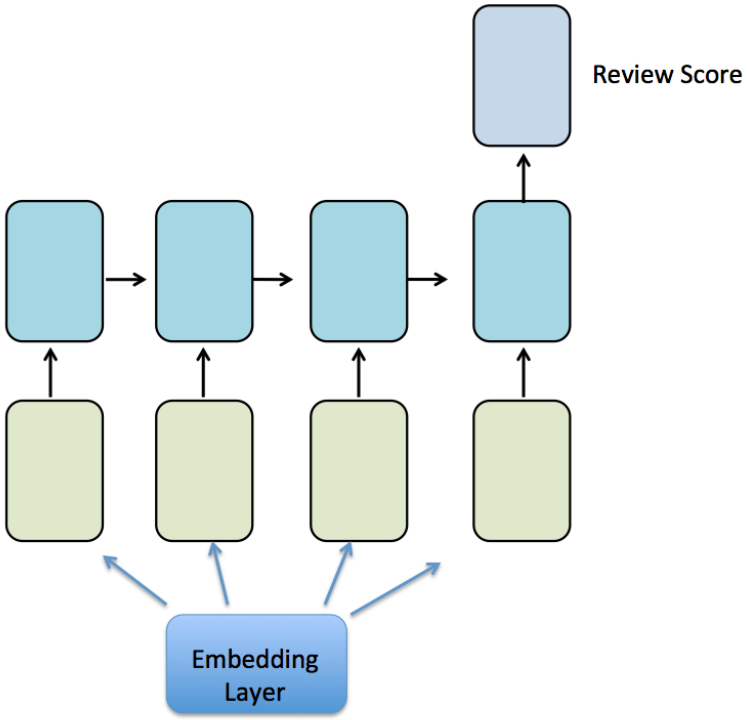



Figure 7: Recurrent network for sentiment analysis

With respect to machine translation, we using recurrent neural networks with sequential output. Figure 8 gives the architecture of a two-layer, recurrent network that can translate sentences from one language to another. A sentence is mapped to a sequence of words and then passed through the network. The network outputs a sequence of words in the second language. The state-of-art machine translation algorithms at Google and Microsoft use architectures similar to this; however, the also have extra heuristics to boost results. We will have a demo of this by the end of 2016.

With the advent of memory networks, question and answer systems have acquired new capabilities. These networks can ingest stories and answer questions that require transitive reasoning. Each sentence in the story becomes a memory or fact. The network learns to chain relevant memories to answer the question. The input sentences can take story form, in which time plays a role, e.g. *"John went to the kitchen. John picked up an apple. John walk to the bedroom. Where is the apple?"*, or not. Researchers are investigating memory networks to enable searches that return the answer

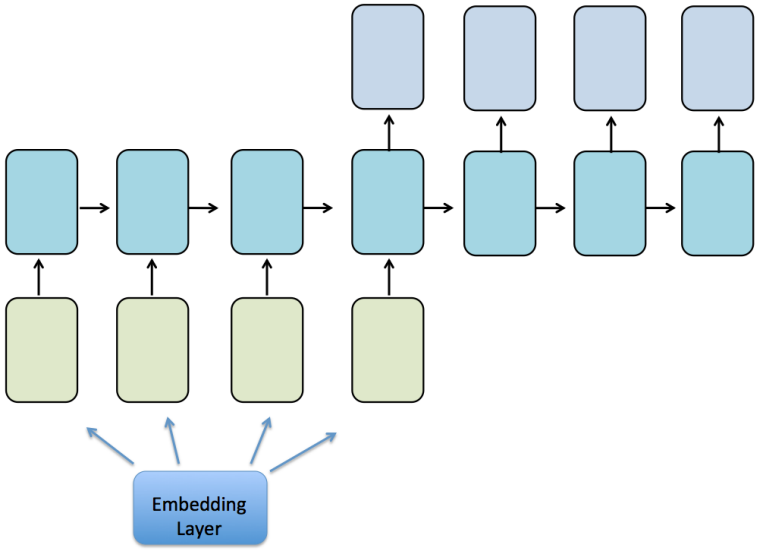



Figure 8: Recurrent network for machine translation

instead of a list of links. We are investigating using memory networks in combination with key-word search. A user will ask a question. The question will be parsed and a key-word search will be performed. The returned documents can be ingested into a memory network and reasoning can begin.

## References


[1] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, 2013.

[2] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1162.

[3] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Stroudsburg, PA, October 2013. Association for Computational Linguistics.

[4] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In C.j.c. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273, 2013. URL http://media.nips.cc/nipsbooks/nipspapers/paper_files/nips26/1097.pdf.

[5] A. Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL http://arxiv.org/abs/1308.0850.

[6] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306, 2014. URL http://arxiv.org/abs/1412.2306.

[7] *Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding*, August 2013. URL http://research.microsoft.com/apps/pubs/default.aspx?id=193771.

[8] C. Raymond and G. Riccardi. Generative and discriminative algorithms for spoken language understanding. In *INTERSPEECH*, pages 1605–1608. ISCA, 2007. URL http://dblp.uni-trier.de/db/conf/interspeech/interspeech2007.html#RaymondR07.

[9] K.S. Tai and R. Socherand C. D.Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *ArXiv e-prints*, February 2015.

[10] T.i Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[11] Emmanuel J. Candès, Yonina C. Eldar, Deanna Needell, and Paige Randall. Compressed sensing with coherent and redundant dictionaries. *Applied and Computational Harmonic Analysis*, 31(1):59 – 73, 2011. ISSN 1063-5203. doi: http://dx.doi.org/10.1016/j.acha.2010.10.002. URL http://www.sciencedirect.com/science/article/pii/S1063520310001156.

[12] E. Candes. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9-10):589–592, May 2008. ISSN 1631073X. doi: 10.1016/j.crma.2008.03.014. URL http://dx.doi.org/10.1016/j.crma.2008.03.014.

[13] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, 2006.

[14] S. Ji, Y. Xue, and L. Carin. Bayesian compressive sensing. *IEEE Trans. Signal Processing*, 56(6):2346–2356, 2008.

[15] T. Strohmer. Measure what should be measured: Progress and challenges in compressive sensing. *CoRR*, abs/1210.6730, 2012. URL http://arxiv.org/abs/1210.6730.

[16] J. A. Tropp. Just relax: Convex programming methods for subset selection and sparse approximation. Technical report, 2004.

[17] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

[18] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 455–465, 2013. URL http://aclweb.org/anthology/P/P13/P13-1045.pdf.

[19] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285, 2015. URL http://arxiv.org/abs/1506.07285.